НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Програмна інженерія’’

на тему: Моніторинг рівнів гідропостів з використанням технології WebSocket

Виконав: студент 4 курсу, групи ТМ-52

Найдьонов Максим Васильович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ас. Швайко Валерій Григорович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент к.т.н., доцент Баранюк О.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “ Комп'ютерні науки’’

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ

на дипломну роботу студенту

Найдьонову Максиму Васильовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “ Моніторинг рівнів гідропостів з використанням технології WebSocket ”
керівник роботи ас. Швайко Валерій Григорович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)
затверджена наказом вищого навчального закладу від ”___”_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.
3. Вихідні дані до роботи файл з розширенням .txt, в якому зберігаються результати моніторингу
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та засоби моніторинг рівнів гідропостів, розробити програмне забезпечення, розробити інтерфейс користувача
_____
_____
_____
_____
5. Перелік ілюстративного матеріалу

1. Мета та завдання роботи 2. Огляд існуючих рішень 3. Характеристика гідропостів 4. Аналіз проблеми розробки 5. Моніторинг гідропостів 6. Використані програмні засоби 7. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”_1_”__грудня____2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	1 Затвердження теми роботи	01.12.2018	
2.	Вивчення та аналіз задачі	02.12.2018 — 19.12.2018	
3.	Розробка архітектури та загальної структури системи	19.12.2018 — 29.12.2018	
4.	Розробка структур окремих підсистем	09.01.2019 — 23.01.2019	
5.	Програмна реалізація системи	09.01.2019 — 19.01.2019	
6.	Оформлення пояснювальної записки	22.01.2019 — 12.03.2019	
7.	2 Захист програмного продукту	17.05.2019	
8.	3 Передзахист	31.05.2019	
9.	Захист	17.06.2019 — 21.06.2019	

Студент _____ _____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи було створення модулю навігаційної системи для моніторингу рівнів гідропостів з використанням технології WebSocket. Реалізовано модуль взаємодії зі сторонніми веб-сервісами, необхідними для функціонування навігаційної системи. Інтерфейс модулю повинен надавати можливість виконання запитів та отримання відповідей зі сторони конкретного сервісу з використанням технології WebSocket.

Записка містить 67 сторінки, 14 рисунків, 1 формула та 10 посилань.

ABSTRACT

The purpose of the work was to create a navigation system module for monitoring the levels of hydroposts using WebSocket technology. The module for interaction with third-party web-services necessary for functioning of the navigation system is implemented. The module interface should provide the ability to execute queries and receive responses from a specific service using WebSocket technology.

The note contains, 67 pages, 14 figures, 1 formulas and 10 references.

Короткий опис програми

Розробленій програмний забезпечення, що дозволяє моніторити рівень води у гідропостах з використанням технології WebSocket. Також реалізовано модуль взаємодії зі сторонніми веб-сервісами Google Map API, необхідними для функціонування навігаційної системи.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СКБД — система керування базами даних;

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

БД — база даних;

SQL (англ. Select query language) — декларативна мова програмування для взаємодії користувача з базами даних;

MS (англ. Microsoft) — багатонаціональна корпорація комп'ютерних технологій.

ВСТУП

Серед різноманіття клієнтських додатків важко знайти такі, що не використовують сторонні веб-сервіси для забезпечення додаткового функціоналу та реалізації архітектури клієнт-сервер. Коли мова йде про віддалене збереження інформації, реалізацію аутентифікації користувачів у системі, побудова серверної частини та взаємодіє клієнту з нею є необхідним елементом будь-якої системи. При цьому важливим аспектом побудови модулю, що виконує функції взаємодії з серверною частиною є створення чіткого зрозумілого інтерфейсу з можливістю надання необхідної інформації про стан надісланих запитів та інформування інших частин системи про їх успішну чи неуспішну обробку.

Повноцінну навігаційну систему неможливо створити без використання потужних сервісів для геокодування та отримання інформації щодо необхідної локації на мапі. Також присутність аутентифікації користувачів та збереження даних з використанням бази даних робить необхідним використання додаткових сервісів, що надають такі послуги.

Розв’язком перелічених проблем є використання сервісів Google для забезпечення повного функціоналу всіх частин програмного продукту. Для відображення мапи була використана бібліотека Google Maps Api, а для функціоналу аутентифікації та передачі даних була використана технологія WebSocket.

1 ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ МОДУЛЮ НАВІГАЦІЙНОЇ СИСТЕМИ ДЛЯ ВЗАЄМОДІЇ З ВЕБ-СЕРВІСАМИ

Метою розробки є реалізація окремого модулю навігаційної системи, що надає можливості взаємодії зі сторонніми сервісами.

Призначення даного модулю є надання чіткого зрозумілого інтерфейсу для використання іншими частинами системи з забезпечення повного функціоналу усіх необхідних можливостей. Модуль містить ряд класів, кожен з яких відповідає за певний сервіс, наприклад взаємодія з віддаленою базою даних, аунтентифікація/регістрація користувачів системи, виконання запиту на передачу даних з гідропостів та отримання різноманітної інформації щодо річок, яка потім може бути використана іншими частинами системи.

Необхідними можливостями, які має забезпечувати модуль, є:

— аутентифікація вже існуючих користувачів;

— регістрація нових користувачів за допомогою електронної адреси та паролю;

— можливість відображення всіх гідропостів;

— надання можливості виведення загальної інформації про гідропост над маркером;

— збереження інформації щодо статистики рівня води у річках та опадів

2 ХАРАКТЕРИСТИКА ГІДРОПОСТІВ

У даній роботі було проведено аналіз гідропостів та розглянуто області їх застосування .

2.1 Гідропост

Гідропост - сукупність обладнання і приборів для гідрологічних вимірів і спостереження за річками, осередками, морями та каналами. Також це місце, яке використовують гідрологи або вчені з охорони навколишнього середовища для моніторингу та випробування наземних вод. Як правило, проводяться гідрометричні вимірювання рівня поверхні води ("стадія") та / або об'ємного розряду (потоку), а також можуть бути зроблені спостереження за біотою та якістю води.

Місце розташування вимірювальних станцій часто зустрічається на топографічних картах. Деякі вимірювальні станції є високоавтоматизованими і можуть включати можливість передачі телеметрії в центральний центр реєстрації даних.

2.2 Типи і розряди гідропостів

В залежності від спостережуваного об'єкта і встановленого обсягу спостереження, гідрологічні дослідження мають визначений тип та розряд:

— гідрологічні пості на річках і каналах - ГП. Діляться на ГП: 1-го (ведуть повний обсяг спостереження) і 2,3-го року;

— озерні гідрологічні пости на озерах і водосховищах – ОГП;

— морські гідрологічні пости на морях – МГП.

2.3 Види спостереження

На гідрологічному посту проводяться наступні види спостережень:

— рівень води на водному об'єкті (всі типи)

— ухил водної поверхні (ГП-1)

— витрата води в річці або каналі (ГП-1)

— температура води (всі типи)

— каламутність води (ДП, ОГП)

— витрата зважених і донних наносів (ГП-1)

— хвилювання (МГП, ОГП)

— рейдові спостереження на акваторіях (ОГП, МГП)

— солоність води (МГП)

— моніторинг забруднення вод (всі типи)

Крім того, частина постів здійснює метеоспостереження: температура повітря, опади, снігозйомки тощо.

2.4 Конструкція

Гідрологічний пост забезпечується геодезичним репером з відомою абсолютною висотою. Прив'язка всіх постових пристроїв ведеться по відношенню до цього реперу.

Репер (від фр. repere - знак, вихідна точка) - знак, що закріплює точку земної поверхні, висота якої відносно початкової рівної поверхні визначена шляхом нівелювання.

Всі пристрої можна об'єднати у дві сукупності: це водомірний пост і гідроствор (витратомірного пристрою, присутні тільки на ДП-1).

Водомірні пости діляться, на:

— рейкові водомірні пости - використовують вертикальну рейку з поділками, зазвичай прикріплену до гідротехнічної споруди (мосту або греблі);

— пальові водомірні - пости використовують ряд палі різної висоти, вбитих в дно;

— сучасні дистанційні пости використовують автоматизовані рівнеміри, що передають відліки на відстань. Показання цих станцій часто доступні через Інтернет;

— передавальні водомірні пости - використовують розмічену мотузку або вимірювальну рулетку з підвішеним вантажем

Крім того пости оснащуються самописами рівня води, мареографа і ухил рейками.

Гідроствори оснащуються пристроями з яких і за допомогою яких здійснюється вимірювання витрати води. Гідростворами обладнані тільки річкові гідрологічні пости 1-го розряду.

Гідроствори бувають:

— мостові - оснащені гідрометричних містками;

— колискові - одне і двохтросові колискові переправи;

— човнові;

— оснащені гідрометричних установками - автоматичними (в Україні установка ГР-64) і напівавтоматичними (ГР-70);

— водозливи.

2.5 Історія гідропостів

Стан рівня води в навколишніх водоймах завжди було дуже важливо для людей, особливо для заняття зрошуваних землеробством, тому вимірами рівня води люди стали займатися з глибокої давнини. Наприклад, на Нілі досі збереглися багато ще давньоєгипетські ніломір, яких було близько 20.

За допомогою цих найдавніших гідрологічних постів вимірювався рівень води в Нілі, передбачалися його розливи і їх величина. Деякі з них представляють собою досить складні споруди. У Середні віки деякі стародавні ніломір були відновлені, побудовані нові. Показники рівня води постійно реєструвалися, і, наприклад, по порті ніломір збереглися дані спостережень з 621 року нашої ери. Вважається, що це найдовший у світі ряд систематичних вимірювань.

Станом на 1989 рік в системі Гідрометслужби України, відповідальної за ведення гідрологічного моніторингу, існувало 475 річкових та бл. 75 озерних гідрологічних постів. Станом на 2012 рік - діє 375 гідрологічних постів.

Висновки до розділу

У даному розділі було оглянуто теоретичні відомості про гідропост, типи і розряди, види спостереження, конструкція гідропосту та сфери застосування, які використовуються в даній області.

3 МОНІТОРИНГ ГІДРОЛОГІЧНИХ ОБ'ЄКТІВ

При розробці програмного продукту важливим чинником є розуміння основних понять моніторингу гідрологічних об'єктів. Також для створення програми моніторингу гідропостів треба дослідити поняття гідрологічний моніторинг.

3.1 Гідрологічний моніторинг

Гідрологічний моніторинг - система комплексних спостережень за станом і антропогенним зміною водних об'єктів з метою оцінки, прогнозу і управління цим станом.

Позначені в останні десятиліття зміни клімату на планеті, екологічно небезпечні технології в с/г, зростаюча потреба в якісних продуктах харчування, обумовлює пошук нових підходів до управління та використання природних ресурсів з метою оптимізації природного середовища, збереження стійкого розвитку.

Стійке функціонування ландшафтів та ефективне використання природних ресурсів при мінімальному збитку зовнішнього середовища вимагає створення систем моніторингу стану зовнішнього середовища. Стосовно до водних об'єктів в якості такої системи виступає система Гідрологічного моніторингу (ГМ).

3.1.1 Структура моніторингу

Структура моніторингу повинна передбачати: способи класифікації, ранжирування і визначення співпідпорядкованості підсистем, блоків і пунктів моніторингу та схеми управління їх інформаційної та виробничої діяльністю; моделі централізованого контролю за структурно-функціональним станом системи моніторингу в цілому та процесами видачі користувачам необхідної інформації на електронному або паперовому носії в зручній для споживачів формі.

3.1.2 Методи моніторингу

Методи моніторингу включають: всі властиві гідрології загальнонаукові методи (системний, математичний, моделювання, картографічний), все конкретно-наукові методи (геофізичний, геохімічний, біогеографічний, економічний, соціологічний та ландшафтний) і групу спеціальних або прикладних методів.

3.1.3 Засоби моніторингу

Засоби моніторингу містять такі складові: логічні - робочі гіпотези, судження, докази, формули; інформаційні - апаратура і пристрої для збору, систематизації, обробки, зберігання та передачі оперативних і фондових даних від підсистем і пунктів моніторингу і для обміну інформацією між ними; технічні - вимірювальні прилади, інструменти та обладнання, необхідні для спостережень і контролю за факторами моніторингу; біологічні - живі організми, що використовуються в якості індикаторів моніторингу.

3.1.4 Режим моніторингу

Режим моніторингу включає: групу операцій, прийомів, процедур і алгоритмів необхідних для спостереження, оцінки і прогнозування чинників і показників моніторингу; характеристики дискретності ведення моніторингу, оперативності, завчасності і довгостроковості одержуваних даних; встановлення періодизації здійснення моніторингу - постійний, тимчасовий (сезонний), епізодичний; обґрунтування виконання фактичних і прогностичних оцінок гідрологічного і гідрохімічного стану водних об'єктів і їх водозборів в регіональному, басейновому або локальному масштабах для заданих показників ГМ. В цілому структура інформаційно-яка аналізувала підсистеми ГМ включає три основні блоки:

1. Спостереження - фіксації фактичних даних про фактори і показники стану водних об'єктів і їх водозборів;

2. Оцінювання - порівняння фактичних даних про стан водних об'єктів і їх водозборів з їх нормативними (заданими) значеннями;

3. Прогнозування - порівняння прогностичних даних про стан водних об'єктів і їх водозборів з їх нормальними (заданими) значеннями;

Вся інформація, що отримується в процесі функціонування ГМ, в подальшому використовується в підсистемі управління водними ресурсами для обґрунтування прийняття рішень з водозабезпечення, захисту території від повеней і паводків, поліпшення якості природних вод і, нарешті, для розробки правил управління водогосподарськими системами річкових басейнів в нормальних і надзвичайних ситуаціях.

3.2 Водний режим річок і характеристика річкового стоку

Вода, що проносяться річками, надходить в них в результаті випадання атмосферних опадів на земну поверхню в процесі кругообігу води на земній кулі. Однак в залежності від конкретних умов надходження атмосферної вологи безпосередньо в річки води, які беруть участь в живленні річок, зазвичай ділять на снігові, дощові, підземні і льодовикові (включаючи вічні сніги). В окремих випадках буває досить важко виділити достатньо чітко роль різних джерел живлення в формуванні сумарного стоку річки; в цьому випадку застосовують термін «змішане харчування». На території Росії основна маса річок (близько 60%) отримує водне харчування за рахунок танення сезонних снігів. У південних степових районах України і деяких інших регіонах, де ґрунтові води залягають глибоко і не дренуються річками, а літні дощі не дають поверхневого стоку, річки цілком харчуються водами, що утворюються навесні від танення снігу.

Фази водного режиму. У режимі стоку річок можна виділити ряд характерних періодів (фаз) в залежності від зміни умов харчування. Стосовно до режиму річок розрізняють наступні фази водного режиму:

1) під час повені;
2) паводки;
3) межень.

Повінь в залежності від умов його формування може бути весняним і літнім або весняно-літнім. Повінь характеризується найбільшою в році (серед інших фаз режиму) водністю, високим і тривалим підйомом рівня, зазвичай супроводжується виходом води з русла на заплаву. Викликається головним джерелом харчування (на рівнинних річках - сніготаненням, на високогірних - таненням снігів і льодовиків, в мусонних і тропічних зонах - випаданням літніх дощів тощо.), І для річок однієї

кліматичної зони щорічно повторюється в один і той же сезон з різною інтенсивністю і тривалістю.

Паводки є швидкі і порівняно короткочасні підйоми рівня води в річці; на відміну від повені, виникають нерегулярно; підняття рівня і витрата води при паводку може в окремих випадках перевищувати рівень і найбільша витрата повені. Виникають паводки в результаті випадання дощів, злив і сніготанення під час зимових відлиг - pppa.ru. До категорії паводків зазвичай відносять щорічне підвищення водності в осінній період в результаті дощів і зменшення випаровування. Ці осінні паводки хоча і повторюються щорічно, але часто не утворюють загальної хвилі і не є настільки значними і регулярними, як повінь.

Межень - фаза водного режиму річки, що характеризується тривалим (сезонним) стоянням низьких (межових) рівнів і витрат води в річці внаслідок сильного зменшення або припинення поверхневого стоку; в цей період річка живиться переважно підземними водами.

Річковий стік. Кількість води, що протікає в річковому руслі за будь-який період часу, називається річковим стоком. Він вимірюється в кубічних кілометрах води у її гирла. Річний стік кожної річки змінюється в залежності від водності року.

Витрата води - це об'єм води, що протікає через дане живий перетин в секунду. Витрата виражається наступною формулою:

$$Q = W * Vср. \quad (3.1)$$

де W- живий перетин, м2; Vср. - середня швидкість течії, м / с.

Живий перетин - площа поперечного перерізу річкового потоку, обмежена рівнем води і змоченим периметром русла. Витрата води характеризує водність річки у даного пункту.

Стік річки розподіляється нерівномірно: розрізняють весняний, літній, осінній та зимовий стоки. Весна для більшості річок - період водопілля.

Висновки до розділу

В даному розділі були розглянуті основних понять моніторингу гідрологічних об'єктів. Для даної програми ми використали поняття гідрологічний моніторинг.

4 АНАЛІЗ ПРОБЛЕМИ РОЗРОБКИ МОДУЛЮ НАВІГАЦІЙНОЇ СИСТЕМИ ДЛЯ ВЗАЄМОДІЇ З ВЕБ-СЕРВІСАМИ

Розглянемо типову архітектуру клієнт-сервер.

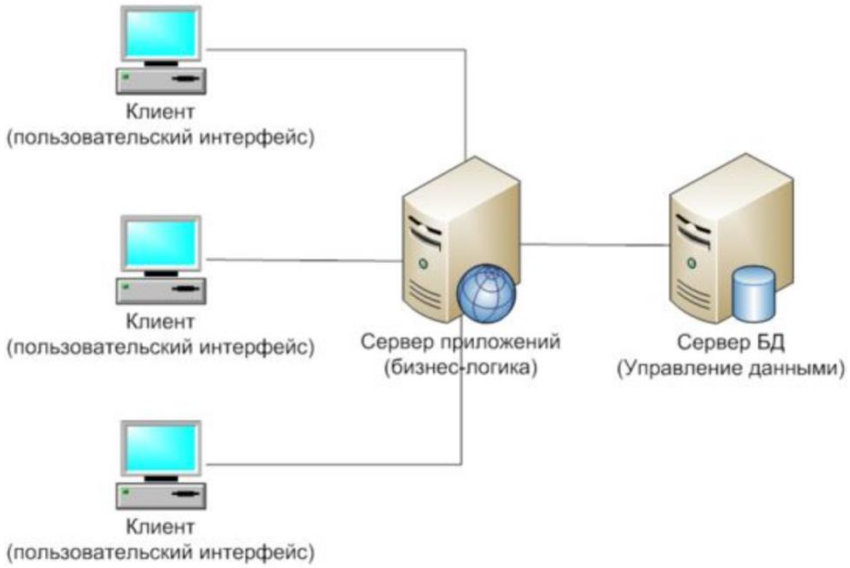

Рисунок 2.1 – Багаторівнева архітектура клієнт-сервер

Кожен клієнт містить реалізацію користувацького інтерфейсу та займається безпосередніми маніпуляціями локальною базою даних. При настанні необхідності виконання певних дій, виконується запит, що містить всю необхідну бізнес-логіку для обробки даних. Додаток, в свою чергу делегує виконання безпосередніх операцій над даними в БД серверу баз даних, що займається виконанням необхідних команд над інформацією, яка знаходиться в базі.

Таким чином, кожен рівень відповідає лише за окремий функціонал та делегує виконання повноважень, які йому не належать, іншому рівню. Це забезпечує можливість паралельної розробки кожного рівня та окремого тестування кожного з них.

4.1.1 ASP .NET MVC

Фреймворк ASP .NET MVC, представляє собою єдиний інтерфейс, що відповідає за функціонал, який за звичайної архітектуру був би розділений між декількома рівнями.

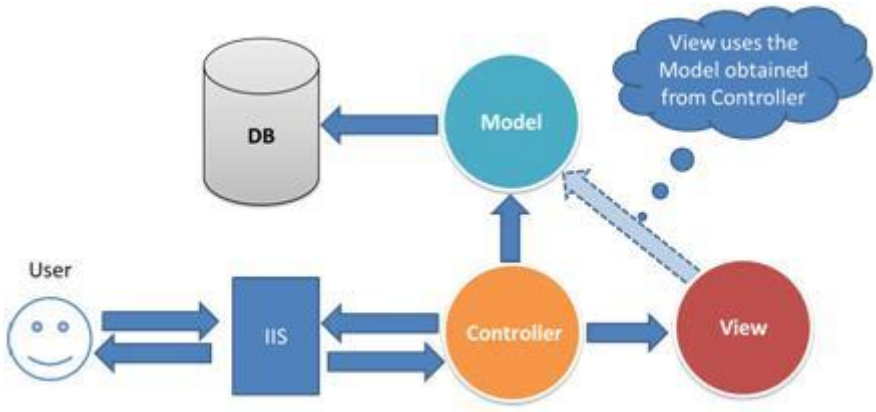

Рисунок 2.2 - Архітектура user-server з використанням ASP .NET

Таким чином, використання ASP .NET MVC виключає необхідність створення окремих рівнів серверної частини для взаємодії з БД, аутентифікацію користувачів і т. д. Надається єдиний інтерфейс, який інкапсулює в собі різноманітний функціонал. Так, ASP .NET MVC є дуже популярним вибором для розробників клієнтських додатків, адже він надає змогу без проектування та написання коду серверної частини використовувати вже готовий функціонал.

До можливостей ASP .NET MVC належать робота з БД, робота з користувачами системи (регістрація та авторизація), відправка повідомлень всім клієнтам з необхідною інформацією і багато інших технологій, які доступні для використання.

Висновки до розділу

В даному розділі був розглянутий фреймворк ASP .NET MVC. Для передачі даних ми використали технологію WebSocket.

5 ЗАСОБИ РОЗРОБКИ

При розробці програмного продукту важливим чинником є правильний вибір технологій та засобів програмної реалізації. Основним середовищем розробки було Microsoft Visual Studio 2017.

Для розробки алгоритмів використовувалась мова C#.

Налаштування додатку для використання даних гідропостів був використаний портал Моніторингу Закарпаття. (http://buvr.bkc.com.ua).

Модуль був розроблений з використанням можливостей мови C#, зокрема бібліотеки Gmap ASP .NET, а для її підключення карти – Google Map API. Для створення інтерфейсу використовувалися Razor.

C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

В якості бази даних була використана MS SQL Server.

5.1 Середовище розробки Visual Studio 2017

Середовище розробки Visual Studio дозволяє швидко і ефективно писати код, не втрачаючи з уваги контекст поточного файлу. Можна легко заглибитися в подробиці, такі як структура виклику, пов'язані функції, повернення і стан тестування. Також доступні рефакторинг коду, знаходження і усунення помилок в коді .

Сьогодні сімейство інструментів Visual Studio 2017 містить IDE, сервіс для організації спільної роботи - Visual Studio Team Services, комплексне рішення для реалізації повноцінного циклу розробки мобільних додатків - Visual Studio Mobile Center, багатоплатформовий редактор коду Visual Studio Code (доступний для Mac, Linux і Windows), а також пробна-версія Visual Studio for Mac.

З кожною версією інструментів Microsoft намагається врахувати побажання розробників і зробити їх зручніше для створення додатків практично для будь-якої платформи. Результатом є величезний інтерес і більше 21 млн установок інструменту на сьогоднішній день.

По-перше, вже зараз абсолютно будь-який розробник може завантажити собі повноцінну версію Visual Studio 2017 і отримати 60-денну безкоштовну передплату для доступу до Xamarin University - навчає сервісу про створення кроссплатформенних мобільних додатків на C #.

По-друге, творці продовжують піклуватися про підвищення продуктивності розробників, створюючи всі умови, щоб сконцентруватися тільки на написанні коду. Наприклад, поліпшення в уже полюбилися можливості навігації за кодом, рефакторінга, виправлення і налагодженні для всіх підтримуваних мов. Додатково, нова версія дозволяє збільшити швидкість командної розробки з новими real-time функцій модульного тестування і перевірки залежностей.

Третя важлива зміна торкнулася процесу установки інструменту. Новітній, полегшений модульний підхід дозволяє вам встановити тільки ті компоненти середовища, які необхідні і прискорює установку інструменту від початку і до кінця.

До того ж, тепер у розробників пропала необхідність створювати проекти і рішення, щоб налагодити будь-який необхідний фрагмент коду.

Останні презентації Visual Studio не обійшлися без демонстрації поліпшень інтеграції з сервісами хмарної платформи Azure. Розробки Microsoft в цьому напрямку дозволяють полегшити створення, налагодження, розміщення і публікацію ваших додатків в хмарі Azure прямо з IDE, надаючи до того ж вбудовані інструменти для роботи цими додатками, а також з Docker-контейнерами, .NET Core додатками і так далі.

Інша важлива зміна на стороні мобільного розробки. Розробники отримали поліпшені інструменти налагодження і профілювання, інструменти генерації модульних тестів. І якщо ви плануєте створювати кроссплатформне додаток, то зараз настав той самий час, коли варто подивитися в бік Visual Studio 2017 і Xamarin, або використовувати альтернативний підхід з Apache Cordova, а можливо і Visual C ++, але вже для створення кроссплтаформенних бібліотек в рамках того ж інструменту - Visual Studio 2017.

Нові можливості додалися і в Visual Studio Mobile Center (Preview). Нагадаємо, що Visual Studio Mobile Center - новий сервіс для мобільних розробників, представлений в середині листопада і створений для того, щоб надати комплексне рішення по збірці, тестування, поширенню і моніторингу мобільних додатків. Сьогодні творці Mobile Center оголосили про підтримку додатків, написаних на Swift, ObjectiveC і Java, разом з Xamarin і React Native додатками, оголошеними раніше. Також з'явилися можливості по створенню Distribution Groups, підтримка Espresso, і поліпшена аналітика. Зараз будь-який бажаючий може спробувати Visual Studio Mobile Center Preview безкоштовно.

З листопада 2016 року, ми продовжуємо стежити за розвитком Visual Studio for Mac. Сьогодні анонсований вже четвертий preview-випуск першої IDE від Microsoft на Mac. На даний момент інструмент сфокусований на мобільній розробці, створенні хмарних рішень і додатків під macOS. З початку листопада була додана підтримка .NET Core проектів, NuGet і постійне поліпшення інструментарію

мобільного розробника. Також команда постійно працює над виправленнями і оптимізацією продуктивності.

Для користувачів Enterprise версії інструменту, команда інженерів додала Redgate Data Tools. Цей функціонал дозволить розробникам включити роботу над базами даних в DevOps цикл і побудувати повноцінний цикл випуску продукту, створюючи додатки і бази даних в рамках одного інструмента.

5.1.1 Пишіть код, ні про що не турбуючись

Середовище розробки Visual Studio допомагає при написанні коду, незалежно від мови, від C #, VB і C ++ до JavaScript і Python, надаючи допомогу в реальному часі.

IntelliSense описує API під час введення, а автоматичне завершення збільшує швидкість і точність роботи. Знайомство з новим API прискорюється завдяки звуженню набору значень за категоріями. Засіб підказки дозволяє перевіряти визначення API. Проблемні місця виділяються знаками тильди, які часто відображаються при введенні.

5.1.2 Навігація в контексті

Буває складно розібратися у великій базі коду. Visual Studio допомагає впоратися з цим, не втрачаючи контекст коду або розмітку, з якої ви почали, за допомогою таких можливостей, як відображення визначення. Удосконалений переходу GoTo дозволяє легко фільтрувати дані і вибирати, які типи елементів шукати.

Пошук всіх посилань спрощує угруповання, фільтрацію і пошук в результатах, а також дозволяє зберігати довільне число наборів результатів. Покращене великомасштабне структурне уявлення панелі прокрутки дозволяє швидко знаходити проблеми, а візуалізація структури завжди підкаже, в якому місці блокової структури коду ви перебуваєте.

Легко переглядайте структуру об'єктів в коді за допомогою огляду внутрішніх об'єктів в браузері рішень і швидко знаходите файли в своєму рішенні.

5.1.3 Розуміння коду

При використанні CodeLens не потрібно залишати код, щоб швидко зрозуміти його структуру викликів і перейти до пов'язаних функцій. Можливості CodeLens на цьому не закінчуються: Прямо з поточної позиції в коді ви можете дізнатися, хто останнім змінив метод, або вдало чи виконуються його тести.

5.1.4 Швидке усунення помилок

Значки лампочок допомагають виявляти і виправляти поширені проблеми коду. Найчастіше це відбувається в реальному часі, поки ви вводите код, що дозволяє швидко відреагувати (наприклад, виконати рефакторинг, реалізувати інтерфейси і ін.) прямо в редакторі.

5.1.5 Всі неполадки в списку помилок

Список помилок - це все, що потрібно для переходу до помилок коду в рішенні і їх виправлення, незалежно від джерела: від компіляції і збірки до проблем аналізу коду. У деяких мовах підтримуються динамічні призначені для користувача аналізатори, що дозволяють виявляти проблеми домену під час введення.

Клацніть посилання на код або натисніть клавішу F1, виділивши помилку, щоб виконати пошук вмісту в Інтернеті, яке допоможе усунути виявлені проблеми.

5.1.6 Легко виконуйте рефакторинг

У міру зростання проекту виникає ймовірність того, що ви робите, реструктуризацією та рефакторингом коду, написаного раніше вами або кимось іншим.

Кілька мов, в тому числі C #, VB, а тепер і C ++, підтримують потужні можливості рефакторинга, включаючи витяг методу і перейменування, за допомогою меню швидких дій в Visual Studio Editor.

5.1.7 Візьміть параметри з собою

Нам відомо, що налаштовувати свою середу саме так, як вам зручно, хочеться тільки один раз. Ми зробили так, щоб ці параметри переміщалися разом з вами, коли ви входите в Visual Studio. Крім того, ми перемістимо посвідчення, використовувані вами для доступу до різних служб для розробників.

5.2 ASP .NET MVC та Razor

ASP.NET MVC Framework - фреймворк для створення веб-додатків, який реалізує шаблон Model-view-controller.

Платформа ASP.NET MVC базується на взаємодії трьох компонентів: контролера, моделі та подання. Контролер приймає запити, обробляє користувача введення, взаємодіє з моделлю і представленням і повертає користувачеві результат обробки запиту.

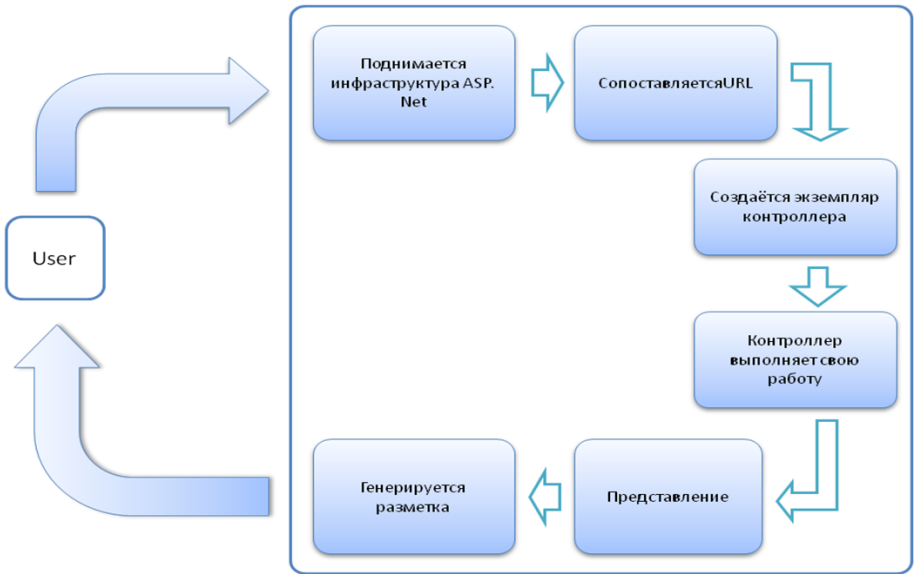

Рисунок 5.1 - Архітектура ASP .NET MVC

Модель представляє шар, що описує логіку організації даних в додатку. Представлення отримує дані з контролера і генерує елементи призначеного для користувача інтерфейсу для відображення інформації.

5.2.1 Генерація уявлень Razor

Для управління розміткою і вставками коду в поданні використовується движок уявлень. До версії MVC 5 використовувалися два движка: Web Forms і Razor.

Починаючи з MVC 5 єдиним двигуном, вбудованим за замовчуванням, є Razor. Движок WebForms використовує файли .aspx, а Razor - файли .cshtml і .vbhtml для зберігання коду уявлень. Основою синтаксису Razor є знак @, після якого здійснюється перехід до коду на мовах C # / VB.NET [26]. Також можливо і використання сторонніх движків. Файли уявлень не є стандартними статичними сторінками з кодом html, а в процесі генерації контролером відповіді з використанням уявлень компілюються в класи, з яких потім генерується сторінка html.

5.2.2 Маршрутизація

При обробці запитів фреймворк ASP.NET MVC спирається на систему маршрутизації, яка зіставляє всі вхідні запити з певними в системі маршрутами, які вказують який контролер і метод повинен обробити такий запит. Вбудований маршрут за умовчанням передбачає триланкову структуру: контролер / дію / параметр.

5.3 Microsoft SQL Server

SQL Server є однією з найбільш популярних систем управління базами даних (СКБД) в світі. Дана СУБД підходить для самих різних проектів: від невеликих додатків до великих високонавантажених проектів [23].

SQL Server був створений компанією Microsoft. Перша версія вийшла в 1987 році. А поточною версією є версія 16, яка вийшла в 2016 році і яка буде використовуватися в поточному керівництві.

SQL Server довгий час був винятково системою управління базами даних для Windows, проте починаючи з версії 16 ця система доступна і на Linux.

SQL Server характеризується такими особливостями як:

— продуктивність. SQL Server працює дуже швидко.

— надійність і безпека. SQL Server надає шифрування даних.

— простота. З даної СУБД відносно легко працювати і вести адміністрування.

Центральним аспектом в MS SQL Server, як і в будь-який СУБД, є база даних. База даних являє сховище даних, організованих певним способом. Нерідко фізично база даних представляє файл на жорсткому диску, хоча таке відповідність необов'язково. Для зберігання і адміністрування баз даних застосовуються системи управління базами даних (database management system) або СУБД (DBMS). І якраз MS SQL Server є однією з такою СУБД.

Для організації баз даних MS SQL Server використовує реляційну модель. Ця модель баз даних була розроблена ще в 1970 році Едгаром Коддом. А на сьогоднішній день вона фактично є стандартом для організації баз даних.

Реляційна модель передбачає зберігання даних у вигляді таблиць, кожна з яких складається з рядків і стовпців. Кожен рядок зберігає окремий об'єкт, а в стовпчиках розміщуються атрибути цього об'єкта.

Для ідентифікації кожного рядка в рамках таблиці застосовується первинний ключ (primary key). В якості первинного ключа може виступати один або декілька стовпців. Використовуючи первинний ключ, ми можемо посилатися на певну рядок в таблиці. Відповідно два рядки не можуть мати один і той же первинний ключ.

Через ключі одна таблиця може бути пов'язана з іншого, тобто між двома таблицями можуть бути організовані зв'язку. А сама таблиця може бути представлена у вигляді відносини ("relation").

Для взаємодії з базою даних застосовується мова SQL (Structured Query Language). Клієнт (наприклад, зовнішня програма) відправляє запит на мові SQL за допомогою спеціального API. СУБД належним чином інтерпретує і виконує запит, а потім посилає клієнту результат виконання.

Спочатку мова SQL був розроблений в компанії IBM для системи баз даних, яка називалася System / R.

У 1979 році компанія Relational Software Inc. розробила першу систему управління баз даних, яка називалася Oracle і яка використовувала мову SQL. У зв'язку з успіхом даного продукту компанія була перейменована в Oracle.

Згодом стали з'являтися інші системи баз даних, які використовували SQL. У підсумку в 1989 році Американський Національний Інститут Стандартів (ANSI) кодифікував мову і опублікував його перший стандарт. Після цього стандарт періодично оновлювався і доповнювався. Останнє його оновлення відбулося в 2011 році. Але незважаючи на наявність стандарту нерідко виробники СУБД використовують свої власні реалізації мови SQL, які трохи відрізняються один від одного.

Виділяються два різновиди мови SQL: PL-SQL і T-SQL. PL-SQL використовується в таких СУБД як Oracle і MySQL. T-SQL (Transact-SQL) застосовується в SQL Server.

5.4 Технологія WebSocket

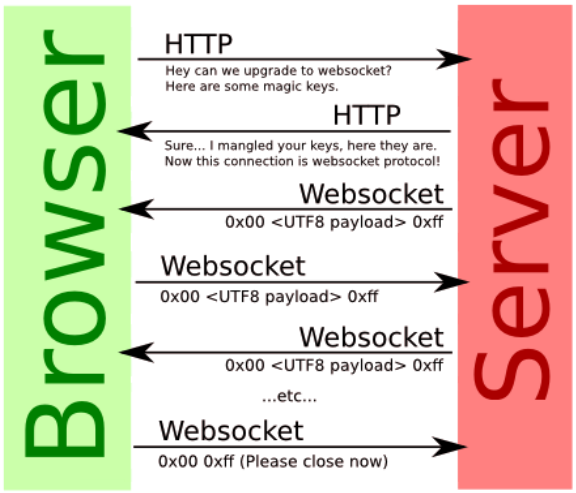

Рисунок 5.2 - Архітектура клієнт-сервер з використанням ASP .NET

WebSocket - протокол зв'язку поверх TCP-з'єднання, призначений для обміну повідомленнями між браузером і веб-сервером в режимі реального часу.

В даний час в W3C здійснюється стандартизація API Web Sockets. Черговий варіант стандарту цього протоколу затверджений IETF.

WebSocket розроблений для втілення в веб-браузерах і веб-серверах, але він може бути використаний для будь-якого клієнтського або серверного додатка. Протокол WebSocket - це незалежний протокол, заснований на протоколі TCP. Він робить можливим більш тісний контакт між браузером і веб-сайтом, сприяючи поширенню інтерактивного вмісту та створення додатків реального часу.

5.5 Карти Google Map Api

Карти Google (англ. Google Maps; раніше Google Local) - набір додатків, побудованих на основі платного картографічного сервісу і технології, що надаються компанією Google.

Сервіс являє собою карту та супутникові знімки планети Земля. Для багатьох регіонів доступні високодеталізовані аерофотознімки (зняті з висоти 250-500 м.), для деяких - з можливістю перегляду під кутом 45 ° з чотирьох сторін світу.

Також там є клієнтські бібліотеки на різних мовах, які дозволяють розробникам використовувати API Google з коду, включаючи Java, JavaScript, Ruby, .NET, Objective-C, PHP і Python.

Доступ до Google Maps API здійснюється за допомогою інтерфейсу HTTP. Для цього застосовуються запити, складені у вигляді рядка URL з використанням текстових рядків або координат широти / довготи для ідентифікації місць, а також ключа API.

Клієнтські бібліотеки полегшують розробку з використанням інтерфейсів API веб-служб Google Maps завдяки простій і ефективної реалізації механізмів

вирішення стандартних завдань, наприклад, аутентифікації, блокування запитів і повторюваного спроб.

Щоб використовувати Google Maps API, потрібно попередньо активувати API в Google API Console і отримати реєстраційні дані для аутентифікації. У кожному запиті потрібно вказувати ключ API (або ідентифікатор клієнта, якщо ви використовуєте преміум-план).

Безпека є важливим фактором, тому при першій-ліпшій можливості рекомендується використовувати протокол HTTPS, особливо для додатків, що містять в запитах чутливі дані користувачів, такі як їх місце розташування. Використання шифрування HTTPS підвищує безпеку додатка, підсилює його захист від крадіжки даних і несанкціонованої модифікації.

Використання Google Map API має певні обмеження:

Service	Limits
Directions	Shared* daily free quota of 100,000 requests per 24 hours; additional requests applied against the annual purchase of Maps APIs Credits. Maximum of 23 waypoints per request. Rate limit applied per user session, regardless of how many users share the same project.**
Elevation	Shared* daily free quota of 100,000 requests per 24 hours; additional requests applied against the annual purchase of Maps APIs Credits. Maximum of 512 points per request. Rate limit applied per user session, regardless of how many users share the same project.**
Geocoding	Shared* daily free quota of 100,000 requests per 24 hours; additional requests applied against the annual purchase of Maps APIs Credits. Rate limit applied per user session, regardless of how many users share the same project.**
Distance Matrix	Shared* daily free quota of 100,000 elements per 24 hours; additional requests applied against the annual purchase of Maps APIs Credits. Up to 100 elements per request (with a maximum of 25 origins and 25 destinations per request). **Note**: requests using `mode=transit` or using the optional parameter `departure_time` when `mode=driving` are limited to 100 elements per request. Rate limit applied per user session, regardless of how many users share the same project.**

* The 100,000 daily free requests are shared across all Maps JavaScript API client-side services and Google Maps APIs web services—all requests are subtracted from the same pool of 100,000 free daily requests. Any additional requests are applied against the total number of Maps APIs Credits you purchased for your Premium Plan. Your free daily request pool is reset at 12:00 am Pacific Time. Note that the 100,000 daily free requests do not apply to Location Services licenses.

** When you first load the API, you are allocated an initial quota of requests. Once you use this quota, the API enforces rate limits on additional requests on a per-second basis. If too many requests are made within a certain time period, the API returns an `OVER_QUERY_LIMIT` response code. The per-session rate limit prevents the use of client-side services for batch requests, such as batch geocoding. For batch requests, use our web service APIs.

Рисунок 5.3 Обмеження на використання сервісу Google Map API

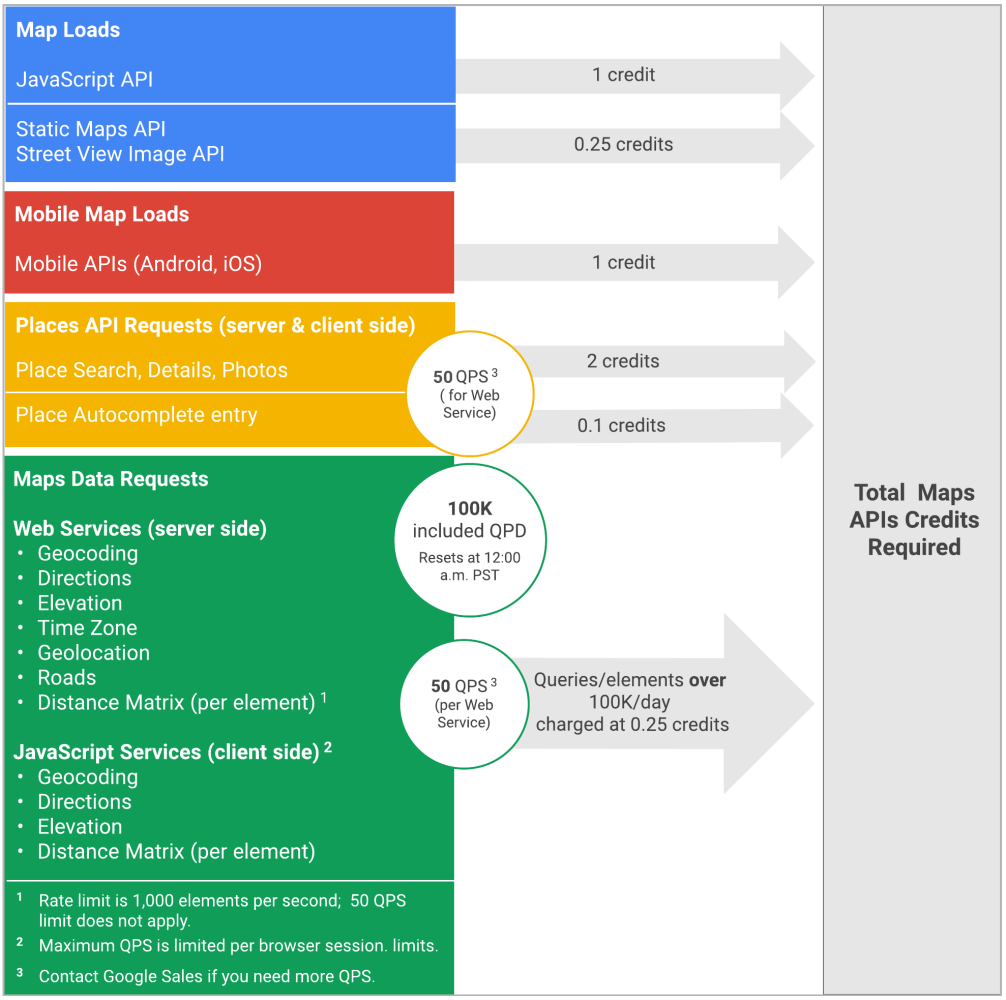

Рисунок 5.4 Обмеження на використання сервісу Google Map API для преміум-плану

Стандартний план для Google Maps API підтримує до 100 000 запитів маршрутів в день. Для проекту буде встановлено обмеження у вигляді безкоштовного щоденного ліміту на 2 500 запитів, поки ви не включите тарифікацію дозволити тарифікацію для проекту.

Google Map Places API Web Service - це служба, з якої за допомогою HTTP-запитів можна отримувати інформацію про місця, визначених у даному API: організаціях, географічні об'єкти або пам'ятки.

Існують наступні основні види запитів, які пов'язані з місцями.

Запити пошуку місць повертають список прилеглих місць, враховуючи місце розташування користувача або умови пошукового запиту.

Запити даних про місце - повертають більш детальну інформацію про конкретне місце, в тому числі відгуки користувачів.

Запити додавання місць дозволяють поповнювати базу Google Map Places інформацією з вашого застосування.

Запити фотографій місць відкривають доступ до мільйонів фотографій, які зберігаються в базі даних Google Map Places.

Підказки місць автоматично пропонують відповідні назви місць або адреси в міру їх введення.

Підказки пошукових запитів забезпечують можливість підказки місць, пропонуючи користувачеві відповідні місця в міру введення текстового запиту по геоданих.

Всі ці служби вживають HTTP-запити і повертають дані в форматі JSON або XML. При цьому в запитах обов'язково повинен використовуватися протокол https: //, а також повинен бути вказаний ключ API.

Обов'язкові параметри:

input - текстовий рядок, по якій виконується пошук. Служба Google Map Places повертає відповідні варіанти на основі цього рядка. Результати будуть відсортовані за релевантністю.

key - ключ API вашого застосування. Цей ключ використовується для ідентифікації додатка з метою управління квотами. Користувачі Google Maps APIs Premium Plan повинні застосовувати проект API, створений для них при покупці Premium Plan.

Додаткові параметри:

offset - положення символу в пошуковому запиті, після досягнення якого служба використовує текст для пошуку схожих запитів. Як правило, значення offset має відповідати положенням текстового курсору. Якщо значення параметра offset не вказано, служба буде використовувати всі умови для пошуку.

location - точка, в околицях якої слід вести пошук інформації про місця. Повинні бути вказані її координати в форматі: широта, довгота.

radius - відстань в метрах, в межах якого повинні знаходитися знайдені результати. Слід враховувати, що при додаванні параметра radius перевага віддається результатами в зазначеній галузі, але оскільки суворе обмеження відсутнє, можуть відображатися результати і за її межами.

language - код мови, на якому слід по можливості повертати результати. Результати пошуку залежать від вибраної мови. Результатам певною мовою може присвоюватися вищий рейтинг. Див. Список підтримуваних мов. Якщо мова не відображається, служба Google Map Places буде використовувати базовий мову домену, з якого надіслано запит.

Висновки до розділу

У даному розділі були розглянуті основні засоби розробки програмного забезпечення. Окрім цього були описані основні бібліотеки та APIs, які використовувалися для забезпечення роботи моніторингу.

6 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Розроблений модуль взаємодії з веб-сервісами представляє собою директорію з класами, що має наступну структуру:

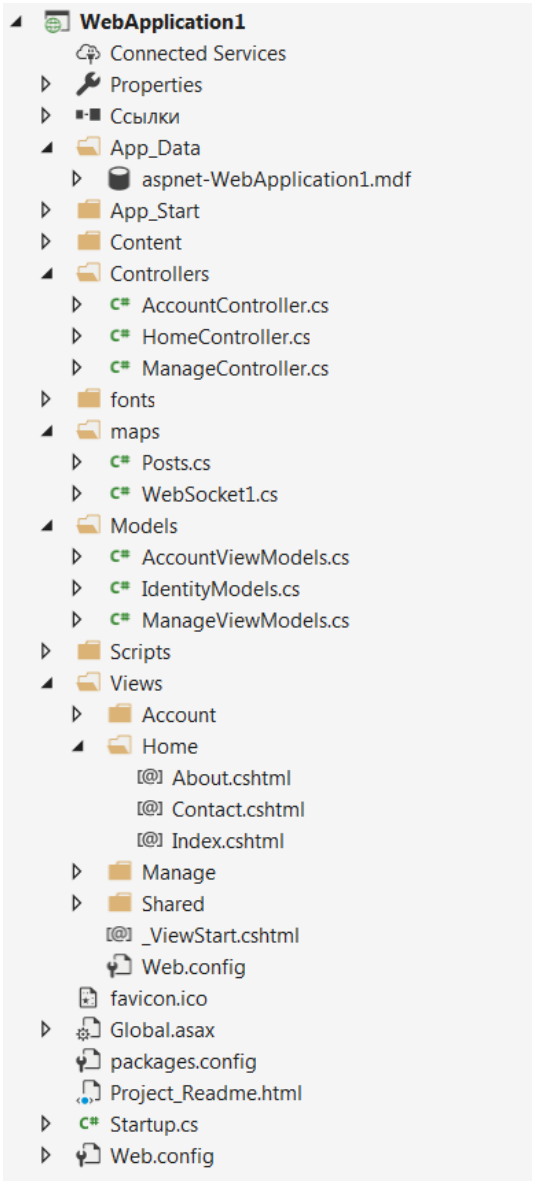

Рисунок 6.1 – Ієрархія модулю

Складовими частинами є піддиректорії Maps та Controllers, що містять відповідні класи для взаємодії з сервісами та класи-моделі, що використовуються для цього відповідно.

6.1 Dependency Injection

При розробці модулю та навігаційної системи в цілому був застосований принцип інверсії управління (inversion of control), також відомий як Dependency Injection.

Впровадження залежності (англ. Dependency injection, DI) — процес надання зовнішньої залежності програмному компоненту, використовуючи «інверсію управління» (англ. Inversion of control, IoC) для розв'язання (отримання) залежностей.

Впроваждення — це передача залежності (тобто, сервісу) залежному об'єкту (тобто, клієнту). Передавати залежності клієнту замість дозволити клієнту створити сервіс є фундаментальною вимогою до цього шаблону проектування.

Існує три найбільш поширені форми впровадження залежностей:

—впровадження в конструктор,

—впровадження у властивість,

—впровадження в метод.

Впровадження залежностей — це шаблон проектування, в якому залежності (або сервіси) впроваджуються, або передаються по посиланню в залежний об'єкт (клієнт) і стають частиною клієнтського стану. Шаблон відокремлює створення залежностей клієнта від власної логіки клієнта, що дозволяє компонентам бути слабко зв'язаними і притримуватися принципів інверсії залежностей і єдиного обов'язку. Це протирічить анти-шаблону service locator, який дозволяє клієнтам знати про систему, що використовується для пошуку залежностей.

Перевагами використання Dependency injection є:

Оскільки впровадження залежностей не вимагає змін у поведінці коду, його можна застосувати як рефакторинг. В результаті цього клієнти стають більш незалежними і над ними легше проводити модульне тестування в ізоляції з використанням макетів об'єкта, які імітують інші об'єкти, від яких залежить об'єкт, що тестується. Простота тестування найчастіше є першою помітною перевагою використання впровадження залежностей.

Впровадження залежностей не вимагає від клієнта знань про конкретну реалізацію, яку йому потрібно використовувати. Це дозволяє ізолювати клієнт від впливу змін проектування і дефектів. Це сприяє повторному використанню, тестуванню і підтримці коду.

Впровадження залежностей може використовуватися для перенесення деталей конфігурації системи в конфігураційні файли, що дозволяє системі змінювати конфігурацію без перекомпіляції. Окремі конфігурації можуть бути написані для різних ситуацій, що вимагають різних реалізацій компонентів.

Впровадження залежностей сприяє паралельній і незалежній розробці. Два розробника можуть незалежно створювати класи, які використовують один одного, знаючи тільки про інтерфейси, через які класи співпрацюють.

Впровадження залежносттей знижує зв'язність між класом і його залежностями.

В даному розділі описується методика подальшого використання розроблених модулів. Описуються робота з системою моделювання акустичного сигналу для дна з постійним кутом нахилу, в якому користувач може задавати початкові параметри моделювання та описується робота з керуючим модулем.

6.2 Приклад роботи карти

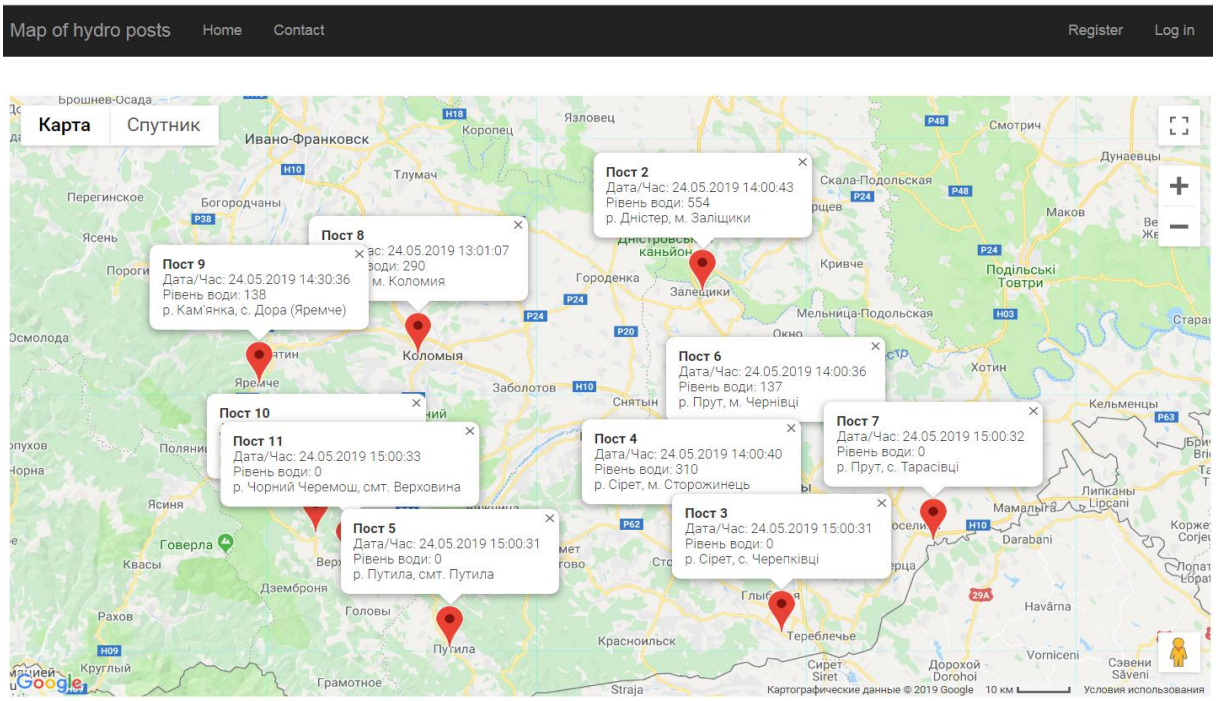

Рисунок 6.2 – Приклад роботи карти

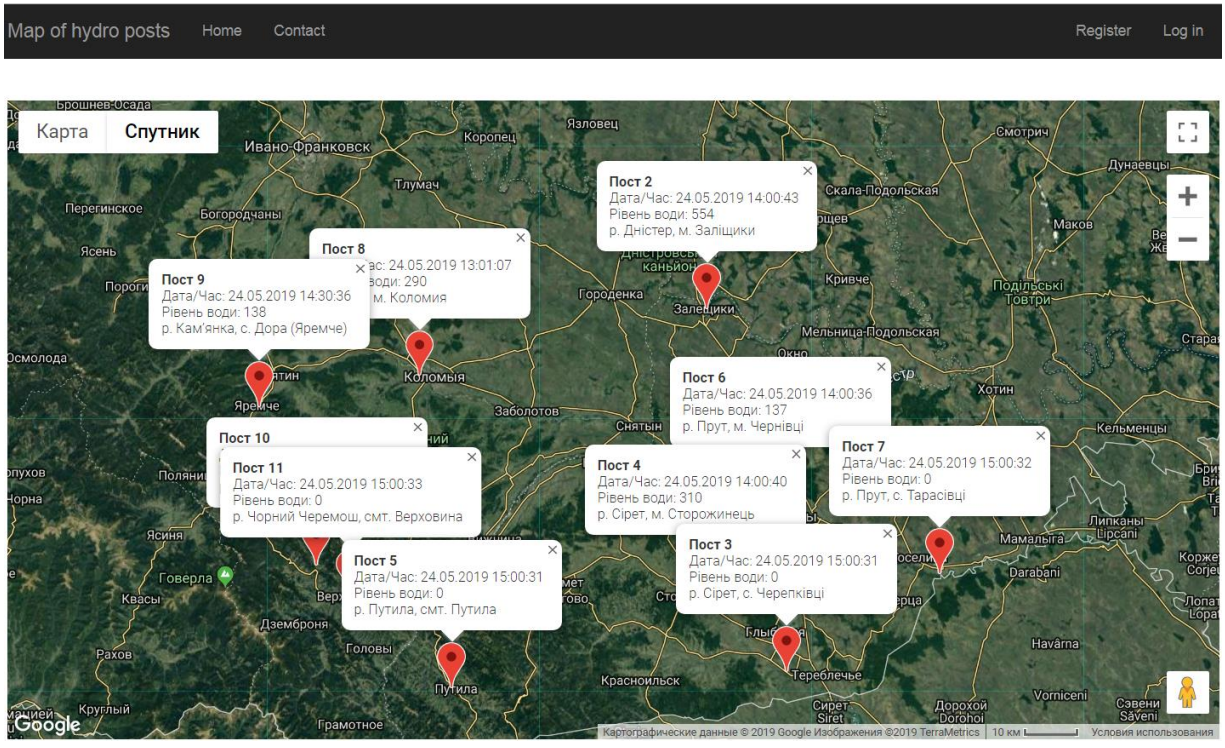

Рисунок 6.3 – Приклад роботи карти зі супутника

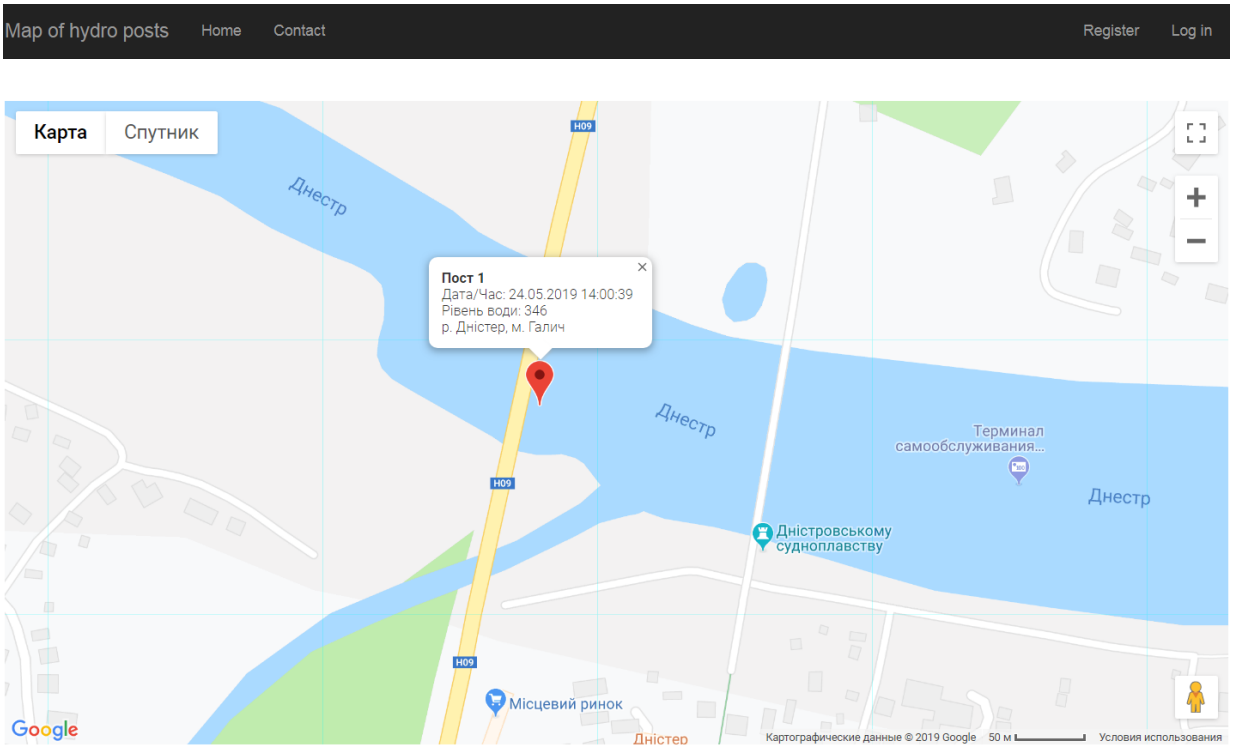

Рисунок 6.4 – Точні координати гідропосту

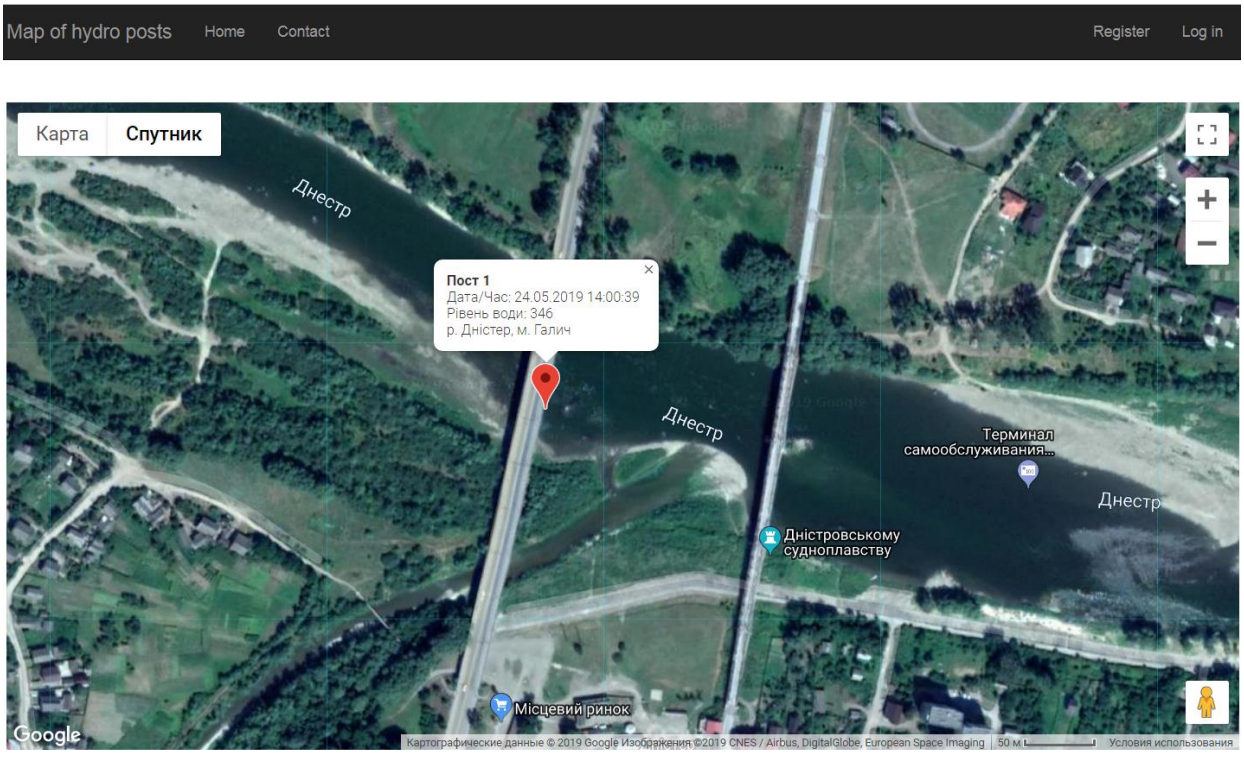

Рисунок 6.5 – Точні координати гідропосту зі супутника

6.3 Основні класи

До класів, що містять бізнес-логіку та забезпечують функціонал взаємодії з тим чи іншим сервісом через технологію WebSocket, належать наступні класи:

WebSocket1.cs – один із класів, що реалізує можливість відправлення запитів за домомогою WebSocket та парсинг отриманих відповідей від порталу Моніторингу Закарпаття. (http://buvr.bkc.com.ua). Даний клас займається безпосередньою відправкою запитів.

Дані передаються у вигляді невеликої кількості стандартному форматі JSON.

JSON будується на двох структурах:

Набір пар ім'я/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом.

Впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

Це універсальні структури даних. Теоретично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Оскільки JSON використовується для обміну даними між різними мовами програмування, то є сенс будувати його на цих структурах.

У JSON використовуються такі їхні форми:

Об'єкт — це послідовність пар ім'я/значення. Об'єкт починається з символу { і закінчується символом }. Кожне значення слідує за : і пари ім'я/значення відділяються комами.

Масив — це послідовність значень. Масив починається символом [і закінчується символом]. Значення відділяються комами.

Значення може бути рядком в подвійних лапках, або числом, або логічними true чи false, або null, або об'єктом, або масивом. Ці структури можуть бути вкладені одна в одну.

Рядок — це послідовність з нуля або більше символів юнікода, обмежена подвійними лапками, з використанням escape-послідовностей, що починаються із зворотної косої риски (backslash). Символи представляються простим рядком.

Тип Рядок (String) дуже схожий на String в мовах C і Java. Число теж дуже схоже на C- або Java-число, за винятком того, що вісімкові та шістнадцяткові формати не використовуються. Пропуски можуть бути вставлені між будь-якими двома лексемами.

Posts.cs – головний клас навігаційної системи. Містить в собі інформацію про точних координати постів, а також сюди передають інформацію щодо стану гідропосту.

Для забезпечення можливості регістрації/авторизаціх користувачів був використаний відповідний клас AccountController. Даний клас містить методи для регістрації нових користувачів за допомогою пари електронна адреса/пароль, а також авторизації вже існуючих користувачів.

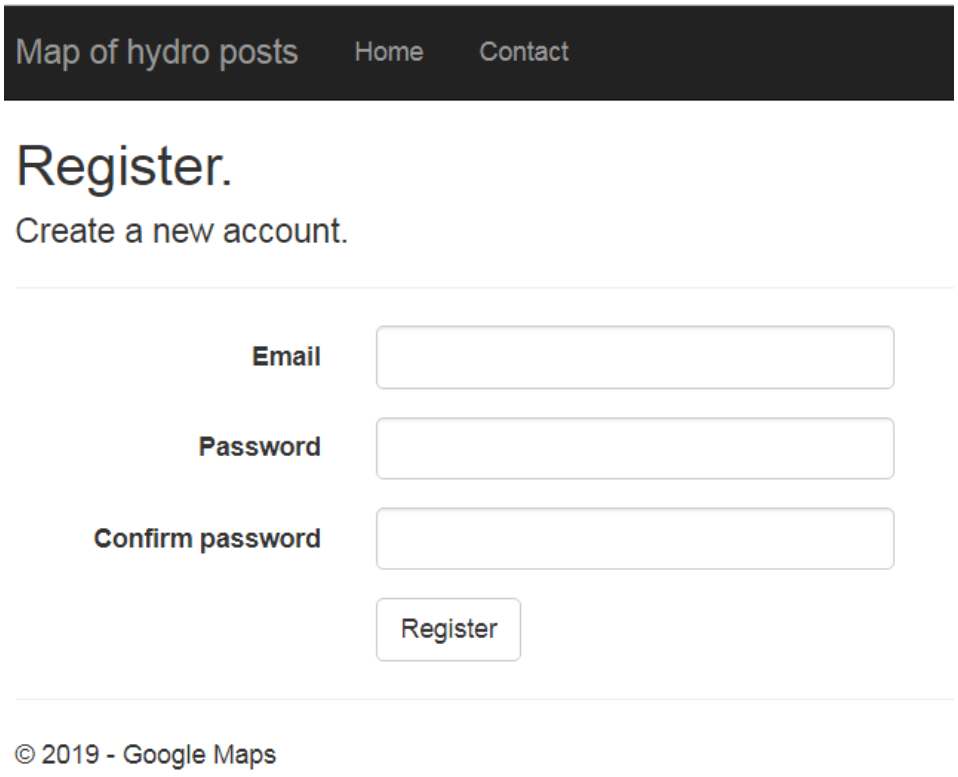

Рисунок 6.6 – Форма регистрації

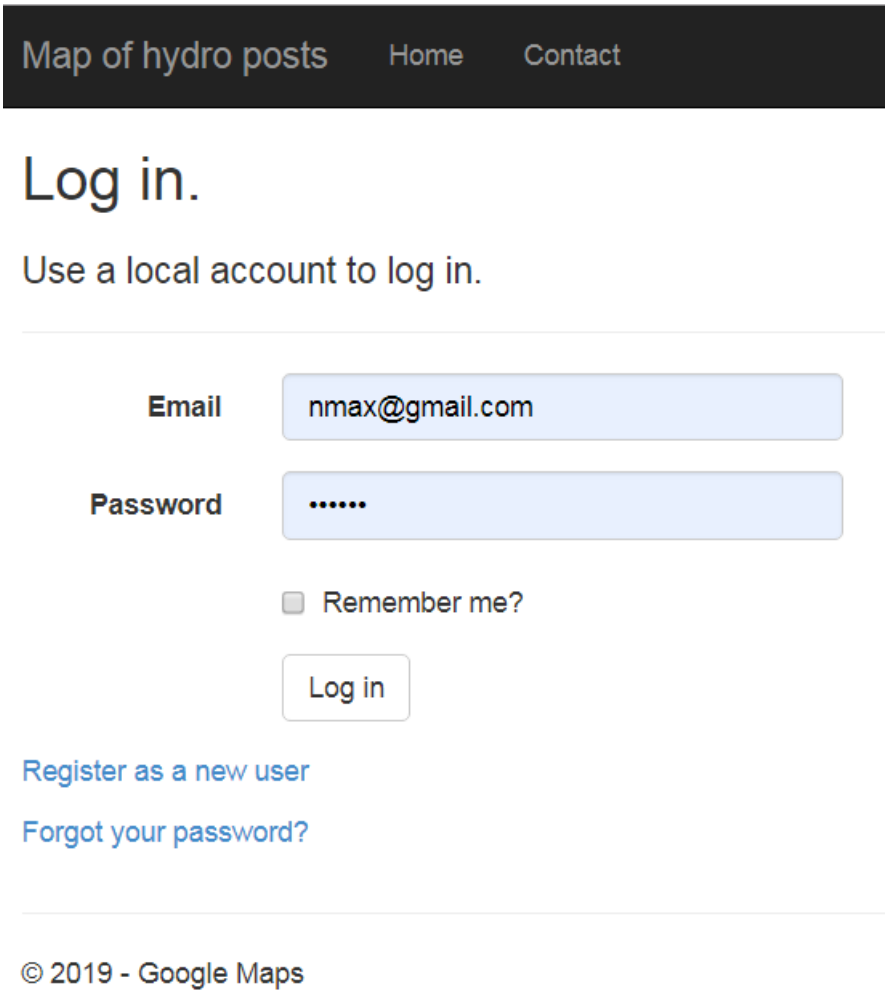

Рисунок 6.7 – Форма входу у систему

6.4 База даних

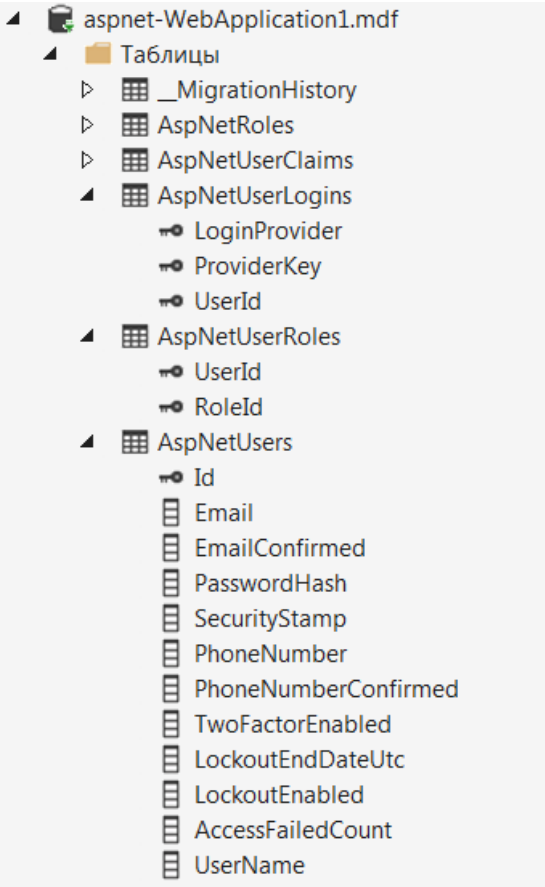

Рисунок 6.8 – Структура БД

БД створюється автоматично за допомогою ASP .NET

БД складається з основної таблиці AspNetUsers.

Таблиця AspNetUsers містить список усіх користувачів системи, кожен користувач містить список свій логін, пароль та загальна інформація. Таким чином встановлюється відповідність між користувачем та даними авторизвції.

Висновки до розділу

В даному розділі були розглянуті розроблені додатки, а також було описано процес їх реалізації та надано інструкції, щодо їх застосування

ВИСНОВКИ

Під час проходження практики було покращено навички розробки прогрмних продуктів за допомогою середовища розробки програмного забезпечення Microsoft Visual Studio 2017 та мови програмування C#.

Також були отримані навички застосування принципів реактивного програмування, архітектури впровадження залежностей та архітектури клієнт-сервер. Було виконано глибоке дослідження сервісу, Google Maps API. Фреймворк ASP .NET забезпечив можливість збереження даних самого користувача. Використання Google Maps Api надало можливість відображення карти та маркерів для представлення загальної інформації про гідропости, з використанням WebSocket.

Таким чином, проходження практики поглибило знання різноманітних технологій розробки клієнтських застосунків та взаємодії зі сторонніми сервісами, використання яких є невід’ємною частиною сучасних програмних продуктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Географічна енциклопедія України : В 3-х т / Редколегія: О. М. Маринич (відповід. ред.) та ін. — К. : «Українська Радянська Енциклопедія» ім. М. П. Бажана, 1989—1993. — 33 000 прим. — ISBN 5-88500-015-8.
2. Загальна гідрологія: Підручник. / за ред. В. К. Хільчевського і О. Г. Ободовського. — К.:, ВПЦ «Київський університет», 2008.
3. Адам Фримен — ASP.NET Core MVC с примерами на C# для профессионалов [Електронний ресурс]. — 2017. — Режим доступу: https://bit.ly/2JbSgHe.
4. 2. Джеймс Чамберс — ASP.NET Core. Разработка приложений [Електронний ресурс]. — 2018. — Режим доступу: https://bit.ly/2AliYL2.
5. Боровков А. И. Компьютерный инжиниринг / А. И. Боровков. — СПб: Изд-во Политехн. ун-та, 2012. — 93 с.
6. Pro C# 7: With .NET and .NET Core, 8th Edition, 2017. – 1372 с.
7. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://www.visualstudio.com/.
8. Microsoft SQL Server [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017.
9. Documents open method [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/ru-ru/library/microsoft.office.interop.word.documents.open.aspx.
10. System Diagnostics [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/ru-ru/library/system.diagnostics(v=vs.110).aspx.

ДОДАТОК А

Моделювання акустичного сигналу для дна з постійним кутом нахилу та керуючий модуль системи.

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ41107_18Б

Аркушів 1

Київ 2018

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТМ52101_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТМ52101_19Б 12-1	Posts.cs WebSocket1.cs AccountController.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТМ52101_19Б 13-1	Додаток В.doc	Опис програмного модуля

ДОДАТОК Б

Моніторинг рівнів гідропостів з використанням технології WebSocket.

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТМ52101_19Б 12-1

Аркушів7

Київ 2018

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.IO;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System.Data.Linq;

namespace HidroCollection
{
class WebSocket1
    {
       private static List<int> hpNums = new List<int>();
       private static List<int> opadyNums = new List<int>();
        private static Dictionary<int, double> limitLevel = new Dictionary<int,
double>();
        [STAThread()]
        static void Main(string[] args)
        {


            string _urlPostNum = "http://buvr.bkc.com.ua/rest/getsensors";

            GetHidroPostFromJson(_urlPostNum);
            GetPrecipitatinPostFromJson(_urlPostNum);
            using (System.IO.StreamReader file =
           new System.IO.StreamReader(@"D:\HidroCollection\HPLimitLevel.txt", false))
            {
                string line;
                while ((line = file.ReadLine()) != null)
                {
                    string[] lines = line.Split(',');
                    limitLevel.Add(Convert.ToInt32(lines[0]),
Convert.ToDouble(lines[1]));
                    Console.WriteLine(line);
                }


            }
```

```
                List<string> hpLevels = new List<string>();
                foreach (int hpNum in hpNums)
                {
                    string                              _url                              =
"http://buvr.bkc.com.ua/rest/getmeasurments?sensorid="+hpNum.ToString()           +"&from="+
DateTime.Now.Day+"-"+ DateTime.Now.Month+ "-"+ DateTime.Now.Year+"&to="+ DateTime.Now.Day +
"-" + DateTime.Now.Month + "-" + DateTime.Now.Year;

                    string                hpLevel=                hpNum.ToString()+";"+
GetHidroMeasureFromJson(_url,hpNum);

                    hpLevels.Add(hpLevel);
                }

                //запись в файл
                using (System.IO.StreamWriter file =
               new System.IO.StreamWriter(@"D:\HidroCollection\ProfilMeasure.txt", false))
                {
                    file.WriteLine("Id;DateM;LevelHP;Excess");
                    foreach (string line in hpLevels)
                    {
                            file.WriteLine(line);
                    }
                }

                List<string> precLevels = new List<string>();
                foreach (int hpNum in opadyNums)
                {
                    string _url = "http://buvr.bkc.com.ua/rest/getmeasurments?sensorid=" +
hpNum.ToString()  +  "&from="  +  DateTime.Now.Day  +  "-"  +  DateTime.Now.Month  +  "-"  +
DateTime.Now.Year  +  "&to="  +  DateTime.Now.Day  +  "-"  +  DateTime.Now.Month  +  "-"  +
DateTime.Now.Year;
                    string     hpLevel     =     hpNum.ToString()     +     ";"     +
GetPrecipitationMeasureFromJson(_url);

                    precLevels.Add(hpLevel);
                }
                //запись в файл
                using (System.IO.StreamWriter file =
               new  System.IO.StreamWriter(@"D:\HidroCollection\PrecipitationMeasure.txt",
false))
```

```
                {
                    file.WriteLine("Id;DateM;LevelHP");
                    foreach (string line in precLevels)
                    {
                        file.WriteLine(line);
                    }
                }
                Console.ReadLine();
            }
            private static string GetHidroMeasureFromJson(string url,int HPnum)
            {
                using (WebClient client = new WebClient())
                {
                    string _dataLevel = DateTime.Now + ";0";
                    client.Encoding = Encoding.UTF8;
                    client.UseDefaultCredentials = true;
                    client.Credentials = new NetworkCredential("rest", "3UgsO7lPX2Rh0");
                    string response = client.DownloadString(url);
                    SortedDictionary<       DateTime,       double>       result       =
JsonConvert.DeserializeObject<SortedDictionary<DateTime, double>>(response);
                    if (result.Count == 0) return _dataLevel;
                    List<DateTime> ldt = new List<DateTime>();
                    foreach(KeyValuePair<DateTime, double> kv in result)
                    {
                        ldt.Add(kv.Key);
                    }
                    ldt.Sort();
                    DateTime lastDate = ldt[ldt.Count - 1];
                    double _level = 0.0;
                    double _limit = 0.0;
                    result.TryGetValue(lastDate, out _level);
                    limitLevel.TryGetValue(HPnum, out _limit);
                    _dataLevel=  lastDate  +  ";"+_level.ToString("f0")  +  ";"  +  (_level-
_limit).ToString("f0");//dd\/MM\/yyyy                    HH:mm:ss                    "hh:mm"
,System.Globalization.CultureInfo.InvariantCulture
                    return _dataLevel;
                }



            }
            private static string GetPrecipitationMeasureFromJson(string url)
```

```
        {
            using (WebClient client = new WebClient())
            {
                string _dataLevel = DateTime.Now + ";0";
                client.Encoding = Encoding.UTF8;
                client.UseDefaultCredentials = true;
                client.Credentials = new NetworkCredential("rest", "3UgsO7lPX2Rh0");
                string response = client.DownloadString(url);
                SortedDictionary<DateTime,          double>          result          =
JsonConvert.DeserializeObject<SortedDictionary<DateTime, double>>(response);
                if (result.Count == 0) return _dataLevel;
                List<DateTime> ldt = new List<DateTime>();
                foreach (KeyValuePair<DateTime, double> kv in result)
                {
                    ldt.Add(kv.Key);
                }
                ldt.Sort();
                DateTime lastDate = ldt[ldt.Count - 1];
                double _level = 0.0;
                result.TryGetValue(lastDate, out _level);
                _dataLevel = lastDate + "; " + _level.ToString("f0");//dd\/MM\/yyyy
HH:mm:ss "hh:mm" ,System.Globalization.CultureInfo.InvariantCulture
                return _dataLevel;
            }


        }
        private static void GetHidroPostFromJson(string url) //async
        {

            using (WebClient client = new WebClient())
            {
                client.Encoding = Encoding.UTF8;
                client.UseDefaultCredentials = true;
                client.Credentials = new NetworkCredential("rest", "3UgsO7lPX2Rh0");
                string response = client.DownloadString(url);
                var     result     =     JsonConvert.DeserializeObject<Dictionary<int,
string>>(response);
                foreach (KeyValuePair<int, string> kv in result)
                {
```

```
                        if (kv.Value.Contains("Рівень води") && !kv.Value.Contains("Рівень
води в")) hpNums.Add(kv.Key);
                    }
                    //запись в файл
                    using (System.IO.StreamWriter file =
                   new
System.IO.StreamWriter(@"D:\HidroCollection\HidroCollection\HidroPost.txt", false))
                    {
                        foreach (KeyValuePair<int, string> kv in result)
                        {
                            if        (kv.Value.Contains("Рівень        води")        &&
!kv.Value.Contains("Рівень води в"))
                            {
                                string line1 = kv.Key.ToString() + ";" + kv.Value;
                                file.WriteLine(line1);
                            }
                        }
                    }
                }
            }
            private static void GetPrecipitatinPostFromJson(string url) //async
            {
                using (System.Net.WebClient client = new WebClient())
                {
                    client.Encoding = Encoding.UTF8;
                    client.UseDefaultCredentials = true;
                    client.Credentials = new NetworkCredential("rest", "3UgsO7lPX2Rh0");
                    string response = client.DownloadString(url);
                    var     result     =     JsonConvert.DeserializeObject<Dictionary<int,
string>>(response);
                    foreach (KeyValuePair<int, string> kv in result)
                    {
                        if  (kv.Value.Contains("Опади")  &&  !kv.Value.Contains("(опади)"))
opadyNums.Add(kv.Key);
                    }
                    //запись в файл
                    using (System.IO.StreamWriter file =
                   new
System.IO.StreamWriter(@"D:\HidroCollection\PrecipitationPostNew.txt", false))//Sensors
                    {
                        foreach (KeyValuePair<int, string> kv in result)
```

```
                        {
                            if                (kv.Value.Contains("Опади")                &&
!kv.Value.Contains("(опади)"))
                            {
                                string line1 = kv.Key.ToString() + ";" + kv.Value;
                                file.WriteLine(line1);
                            }
                        }
                    }
                }
            }
        }
```

ДОДАТОК В

Моніторинг рівнів гідропостів з використанням технології WebSocket.

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТМ52101_19Б 13-1

Аркушів8

Київ 2019

АНОТАЦІЯ

Розділ містить опис частини, яка слугує для роботи з БД, що є структурною одиницею програмного продукту, та забезпечує поєднання можливостей усіх інших модулів для виконання поставлених перед системою завдань. Призначенням БД є зберігання інформації. Модуль надає можливість отримувати необхідні дані, оновлювати та видаляти необхідні записи. Модуль написано мовою програмування C#, з використанням технології MS SQL Server.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку розглядається один з програмних модулів системи — модуль для роботи з використанням технологією WebSocket з кодом УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ52101_19Б 12-1, що міститься у файлі WebSocket1.cs. Модуль призначений для управління підсистемами, які відповідають за передачу рівнів води гідропостів. Користувач має можливість перегляду необхідної інформації щодо стану річок.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Призначенням модулю для роботи з БД є збереження інформації для клієнтської частини та безпосереднього прийняття даних введення та обмін інформацією із клієнтським інтерфейсом. Використання такого шаблону дозволяє створювати програмне забезпечення, де інтерфейс і логіка роботи модуля для БД є незалежними компонентами, що дає можливість використовувати його для зменшення навантаження на клієнтську частину системи.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Слідкувати за зміною інформації про підсистеми є головним завданням модуля. При запуску системи модуль повертає всю інформацію, яка міститься в БД, для відображення даних на користувацькому інтерфейсі. Також модуль оброблює інформацію, надаючи змогу додавати запис, змінювати або видаляти необхідний.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Модуль розроблено у середовищі розробки Microsoft Visual Studio 2017, що забезпечує набір сервісних функцій та графічний діалог з користувачем, на комп’ютері, що використовував операційну систему Windows 7.

В якості СКБД було використано MS SQL Server.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програмний модуль реалізований як окремий клас, який забезпечує існування клієнтської частини та бізнес-логіки окремо від одного, але разом із цим запуск обох компонентів відбувається одночасно.

Для використання даного модулю не потрібно ніяких дій, оскільки він автоматично спрацьовує після запуску клієнтського додатку.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для модуля є інформація, яку користувач вводить в додатку.

Вихідними даними програмного модуля є необхідні записи, які потрібні для користувача.